Pliny Verification and Validation Guide

Introduction

This document outlines the verification and validation procedures for the Pliny system, which analyzes and visualizes Wikipedia metadata trends. It establishes protocols for ensuring that the system functions correctly, produces accurate data, and meets operational requirements.

System Component Verification

1. Data Pipeline Verification

1.1 Environment Setup Verification

- **Python Version**: Confirm Python 3.11+ installation

  ```
  python --version
  ```

- **Virtual Environment**: Verify venv creation and activation

  ```
  python -m venv .venv
  source .venv/bin/activate  # On Unix/macOS
  ```

- **Dependencies**: Validate all required packages are installed

  ```
  pip install -r requirements.txt
  pip list | grep -E "requests|google-cloud"
  ```

- **Credentials**: Verify BigQuery credentials are properly configured

  ```
  ls ./secrets # Should contain: pliny_bigquery_service_account.json
  ```

1.2 Data Extraction Validation

- Execute minimal data extraction to verify API connectivity:

  ```
  python ./data-pipeline/main.py --ingest
  --ingest-start 2023-01-01 --ingest-end 2023-01-01
  ```

- This should run successfully, and finish with `Data Ingestion for 2023/01/01 finished`.

- This should upload data to BigQuery, you may want to check that the data was correctly uploaded to BigQuery.

1.3 Data Processing Validation

- Run data transformation routines on ingested data:

```
python ./data-pipeline/main.py --score
--score-start 2023-01-01 --score-end 2023-01-01
```

- This should run successfully, and insert final processed data into BigQuery tables.

2. Backend Verification

2.1 Environment Setup Verification

- **Go Version**: Confirm Go 1.23.3+ installation

  ```
  go version
  ```

- **Dependencies**: Verify all dependencies are installed

  ```
  cd backend
  go get .
  go list -m all
  ```

2.2 API Endpoint Validation

- Start backend server locally:

  ```
  go run *.go
  ```

- Test the backend correctly returns data on endpoints:

  ```
  curl "http://localhost:8080/availableDates"
  ```

- Verify the response returns a 200, and a JSON with all ingested dates. Note that this will require that the data ingestion script has already ingested some dates

2.3 Performance Validation

- Execute load testing to verify response times under various loads
- Ensure that responses from BigQuery are cached. Subsequent duplicate requests should be significantly faster than the first request. Note that the default TTL on the cache is 1 hour.

3. Frontend Verification

3.1 Environment Setup Verification

- **Node.js/npm**: Confirm versions, `node` should be 18+, `npm` should be 10+

  ```
  node --version
  npm --version
  ```

- **Dependencies**: Verify all packages install without conflicts

  ```
  cd frontend
  npm i
  ```

3.2 Build Verification

- Verify production build completes without errors:

  ```
  npm run build
  ```

- Check build artifacts for expected structure and files

3.3 Functional Verification

- Run development server:

  ```
  npm start
  ```

- Verify core visualizations render correctly
- Test responsive behavior across various viewport sizes
- Validate all interactive elements function as expected

System Integration Validation

1. End-to-End Data Flow

- Execute complete data pipeline process with a controlled dataset
- Verify backend can successfully query and retrieve the processed data
- Confirm frontend correctly displays the retrieved data
- Validate data consistency across all system components

2. API Contract Validation

- Verify backend API responses match frontend expectations
- Test all endpoints with variety of parameters (different dates, limits, etc)
- Validate error handling and graceful degradation (test invalid dates, large limits, etc)

3. Performance Validation

- Measure response times for complete data visualization workflows, should be < 2 seconds.
- Validate memory and CPU usage on backend VM. Should be $< 50\%$ CPU and Memory usage.

Deployment Validation

1. Pre-deployment Checklist

- Verify all environment variables are correctly set
- Confirm database credentials and connection strings
- Validate server configuration (NGINX, routing, etc.)

2. Deployment Verification

- After backend deployment, verify running process and port accessibility
- After frontend deployment, verify NGINX is serving static files correctly
- Test public URL access and functionality

3. Post-deployment Validation

- Verify all core visualizations are functioning correctly
- Validate backend VM analytics are within expectations.

Data Validation

1. Source Data Validation

- Verify Wikipedia API data matches expected schema
- Verify Wikipedia dumps are uploaded correctly every day, within appropriate time ranges.

2. Processed Data Validation

- Verify transformation logic preserves data integrity. For example, pages with many edits in our data should have actually received those edits (can verify by manually going to Wikipedia for sample pages)
- Validate statistical calculations and aggregations
- Test dataset completeness and coverage of expected time periods. Can do this by querying `availableDates` endpoint on backend (or running corresponding SQL query manually).

3. Visualization Validation

- Verify visualizations accurately represent the underlying data
- Test edge cases such as sparse data periods or anomalous spikes

Documentation Validation

1. Verify README instructions are complete and accurate
2. Validate API documentation matches actual implementation
3. Confirm deployment procedures are comprehensive and current

Continuous Improvement

The verification and validation procedures should be reviewed and updated regularly to incorporate: - New data sources or API changes from WikiMedia - Additional visualization types or analysis methods - Performance optimizations and scaling enhancements - Security improvements and vulnerability mitigations